

ConTExT

The background of the page is a dense, overlapping pattern of stylized document icons. Each icon is a rectangle with a white border and a gray fill, containing three horizontal lines representing text. The icons are scattered across the page, creating a textured, layered effect.

title : ConTExT Environment File
subtitle : Homework assignments in ConTExT
author : Aditya Mahajan
date : November 14, 2006


```
1 \startenvironment env-assign
```

This is the environment file to prepare homework assignments in CONTEX_T. See accompanying Prac_T_E_X article for an introduction on how to use it.

I use US letter paper and resonable margins.

```
2 \setuppapersize[letter][letter]
```

```
3 \setuplayout[
    width=middle,
    height=middle,
    location=middle,
    topspace=1in,
    bottomsace=1in,
    backspace=1in,
    cutspace=1in,
    leftmargin=0in,
    rightmargin=0in,
    leftmargindistance=0in,
    rightmargindistance=0in,
    header=0.5in,
    footer=0.5in,
    headerdistace=0in,
    footerdistance=0in,
]
```

The next commands setup defaults for whitespaces and blanks.

```
4 \setupwhitespace [small]
   \setupblank      [medium]
```

I wanted a slightly informal look so choose Palatino as a text font and Euler as the math font. For the source code listings in the solutions, I choose latin modern as the monotype font.

```
5 \startMPenvironment[global]
   \definetypface
     [MyFace] [rm] [serif] [palatino] [default] [encoding=texnansi]
   \definetypface
     [MyFace] [tt] [mono] [modern] [default] [encoding=texnansi,rscale=1.1]
   \definetypface
     [MyFace] [mm] [math] [euler] [euler] [encoding=texnansi,rscale=1.03]
6 \setupbodyfont [MyFace, 11pt]
   \stopMPenvironment
```

The font definitions are declared inside a MPenvironment so that the metapost graphics, which were used for diagrams, also use the same fonts as the main text.

The output is in color and I chose colors which also print reasonably on a black and white printer.

```
7 \setupcolors [state=start]
   \definecolor [colorone] [r=0.625,g=0,b=0] %dark red
   \definecolor [colortwo] [b=0.625,g=0,r=0] %dark blue
```

`\assignment` Various information about the assignment is passed as

```
\assignment[
  title=...,
  course=...,
  assigned=...,
  due=....]
```

The values passed to `\assignment` are stored as parameters of `Assign`.

```
8 \def\assignment[#1]
  {\getrawparameters
   [Assign]
   [ title=
     course=,
     assigned=,
     due=,
     #1]
   \title{\Assigntitle}}
```

The `\title` takes care of setting up the assignment title. There are probably better ways to do this!

```
9 \definefont [BigFontOne] [RegularSlanted sa 2.5]
\definefont [BigFontTwo] [Regular sa 1.5]

10 \setuphead[title]
  [ style=\BigFontOne,
    command=\assignmenttitle,
    before={\resetnumber [PROBLEM] \setuppagenumber [number=1]},
    after={\blank[big] \bgroup \colortwo
           Assigned on:
           \expanded{\date [\Assignassigned]}
           \hfill
           Due on:
           \expanded{\date [\Assigndue]}
           \egroup\blank},
  ]
```

I wanted the a frame with a shadow around the title. So I define a overlay and use it as a background in `\assignmenttitle` below.

```
11 \startuniqueMPgraphic{shadow}
    fill OverlayBox shifted (3pt,-3pt) withcolor .8white ;
    fill OverlayBox withcolor white ;
    draw OverlayBox withcolor blue ;
    setbounds currentpicture to OverlayBox ;
\stopuniqueMPgraphic

12 \defineoverlay [shadow] [\uniqueMPgraphic{shadow}]
```

This macro does the actual typesetting of the title

```
13 \def\assignmenttitle#1#2%
  {\framed[
    width=broad,
    frame=off,
```

```

        align=middle,
foregroundcolor=colortwo,
    % I want a shadow around the title frame
    background=shadow]
    {\#2\{\colorone\BigFontTwo\Assigncourse}}

```

Next I setup the headers and footers. I want the header and footers to be colored

```

14 \setupheader [text] [color=colortwo]
    \setupfooter [text] [color=colortwo]

```

the assignment title to be shown in the header

```

15 \setupheadertexts [title]

```

say “Solutions” while generating solutions

```

16 \startmode[solution]
    \setupheadertexts [Solutions] []
    \stopmode

```

show page number in middle of footer

```

17 \setupfootertexts [pagenumber]

```

show assigned and due dates in the footer

```

18 \setupfootertexts
    [Assigned: \expanded{\date[\Assignassigned}}]
    [Due: \expanded{\date[\Assigndue}}]

```

On the first page, hide the header and only show page number in the footer.

```

19 \definertext [title] [footer] [pagenumber]
    \setuphead [title] [header=high,footer=]

```

Now, I need to take care of the *body* of the document. First I define the `labeltext` for problem, solution and points.

```

20 \setuplabeltext [problem=Problem,solution=Solution ]
    \setuplabeltext [point=point,points=points]

```

Note the extra space after solution. This is because I use `width=fit` in `\defineenumeration[solution]` later.

`Labeltext` provide flexibility for the actual words used in the assignment. This way, if I want to change problem to question, I can simply do

```

\setuplabeltext [problem=Question]

```

First I define an enumeration called `PROBLEM`. This does bulk of the work, but is hidden from the user.

```

21 \defineenumeration
    [PROBLEM]
    [
        text={\labeltext{problem}},
        location=hanging,
        headstyle=\sc,
        headcolor=colorone,
    ]

```

Homework assignments in CONTEX

```

    before={\resetnumber[formula]\page[desirable]},
    after=\blank,
]

```

`\startprob..` Now, I define the *real* `\startproblem`. This takes care of references and points of each problem. The following syntax is allowed

```

    \startproblem           % No tag or points
    \startproblem [tag]    % Just the tag
    \startproblem {points} % Just the points
    \startproblem [tag] {points} % Both the tag and the points

22 \def\startproblem%
    {\dosingleempty\dostartproblem}

23 \def\dostartproblem[#1]%
    {\startPROBLEM[#1]\dosinglegroupempty\dodostartproblem}

24 \def\dodostartproblem#1%
    {\iffirstargument
    % Check if #1 = 1, use point, else use points
    \bgroup \colortwo (#1 \dopoints{#1}) \group
    \fi}

25 \def\dopoints#1%
    {\doifelse{#1}{1}{\labeltext{point}}{\labeltext{points}}}

26 \def\stopproblem%
    {\stopPROBLEM}

```

Next I setup itemize for the display of parts

```

27 \setupitemize [each] [left=(,right=),stopper=,color=colorone]
    \setupitemize [1] [intro]
    \setupitemize [joinedup,packed]

```

Use characters (a,b,...) at the first level and roman numerals (i, ii, ...) at the second level.

```

28 \setupitemize [1] [a]
    \setupitemize [2] [r]

```

`\startsolu..` In the default mode, the solution environment should gobble its contents. I copy the definition of `\starthiding` to *hide* the solutions.

```

29 \definebuffer [solution]
    \setupbuffer [solution] [local=yes]

```

In solution mode, I redefine solution environment as an enumeration without a number. There are probably better ways to define this environment, but why reinvent the wheel?

```

30 \startmode[solution]
    \defineenumeration
        [solution]
        [
            text=\labeltext{solution},
            number=no,
            headstyle=bold,
            headcolor=colortwo,

```

```

location=serried,
width=fit,
before=\startsolutionbackground,
after=\stopsolutionbackground
]

```

The `exam.cls` class in L^AT_EX allows you to put a frame around the solution and the frame breaks across pages. I wanted to put a frame around the solution, but something more informal than a rectangular frame. CONTEX's manual gave hints on getting different kinds of frames, and I finally settled for a randomized framed. The frame should break around pages, so I used `\textbackground`.

```

31 \definetextbackground
[solutionbackground]
[
mp=background:random,
location=paragraph,
rulethickness=1pt,
framecolor=colorone,
width=local,
leftoffset=1em,
rightoffset=1em,
before={\testpage[3]\blank[2*big]},
after={\blank[2*big]}
]

```

The background in `\textbackground` is drawn using a `useMPgraphic` defined set by `mp=`. I define a `useMPgraphic` `background:random` to get the frame that I wanted. The code looks a bit complicated, because `\textbackground` is supposed to work in a multi-column documents.

```

32 \startuseMPgraphic{background:random}
path p;
for i = 1 upto nofmultipars :
p = (multipars[i]
topenlarged 10pt
bottomenlarged 10pt) randomized 4pt ;
fill p withcolor lightgray ;
draw p withcolor \MPvar{linecolor}
withpen pencircle scaled \MPvar{linewidth};
endfor;
\stopuseMPgraphic

```

An earlier attempt that draws a frame which is squeezed in the center. To use this change `mp=background:random` to `mp=background:squeezed` in above.

```

33 \startuseMPgraphic{background:squeezed}
path p;
for i = 1 upto nofmultipars :
p = multipars[i]
topenlarged 10pt
bottomenlarged 10pt
squeezed 5pt
randomized 1pt ;
fill p withcolor lightgray ;
draw p withcolor \MPvar{linecolor}
withpen pencircle scaled \MPvar{linewidth};

```

```

    endfor;
\stopuseMPgraphic

```

By default, the `textbackground` extends till the page boundary. This does not look good if a page break occurs when there is not enough material to fit in the page. So, we want to limit `textbackground` till the typeset material. This has not been interfaced yet, so I use a low level T_EX command.

```

34 \chardef\kindofpagetextareas\plusone
\stopmode

```

I also wanted an English rule at the end of each assignment. CONTEX_T already defines an English rule. I load `meta-txt.tex` to use it.

```

35 \useMPLibrary [txt]

```

I want the English rule to match the color scheme of the entire document.

```

36 \setupMPvariables
    [EnglishRule]
    [color=colortwo,
    width=0.5\textwidth]

```

I redefined `\stopcomponent` to add the rule at the end of the file.

```

37 \let\normalstopcomponent\stopcomponent
38 \def\stopcomponent%
    {\dosomebreak\nobreak
    \framedtext[frame=off,strut=no,align=middle,width=\textwidth]
    {\EnglishRule}
    \normalstopcomponent}
39 \stopenvironment

```